Digital Logic Design

[2EC303]

Special Assignment

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A' GRADE

-: Topic :-

Elevator Weight Manager

-: Made by :-

Dhruv Dholariya – 20BEC024

Dhruvi Patel – 20BEC026

Introduction to Project

What is Elevator (Lift)?
An elevator can be defined as an electric lift which is used for vertical transportation of goods as well as people among the floors in buildings. These are activated with the electrical motors that also to drive counterweight system cables for drive transaction such as a hoist.
These are used in many areas like agriculture manufacturing, etc. Elevators are classified into different types based on our requirement. Elevators are frequently used in the latest multi-storey constructions.

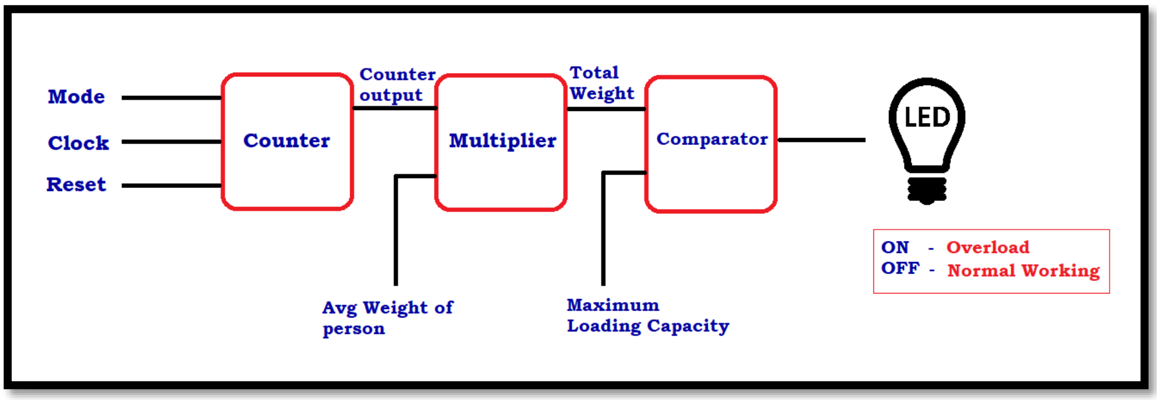
The Components used in the Project :-

1. **4 Bit Mod Counter**
   It is used to count the number of Passengers entering the elevator .

2. **4 Bit Multiplier**
   It is used to calculate the total weight of passengers present in the elevator or it is used to find Total weight inside elevator.

3. **8 Bit Comparator**
   It is used compare the total weight in elevator and the maximum weight capacity of the elevator. If the total weight is greater than total weight then LED will glow indicating Overload condition.

Working of Project

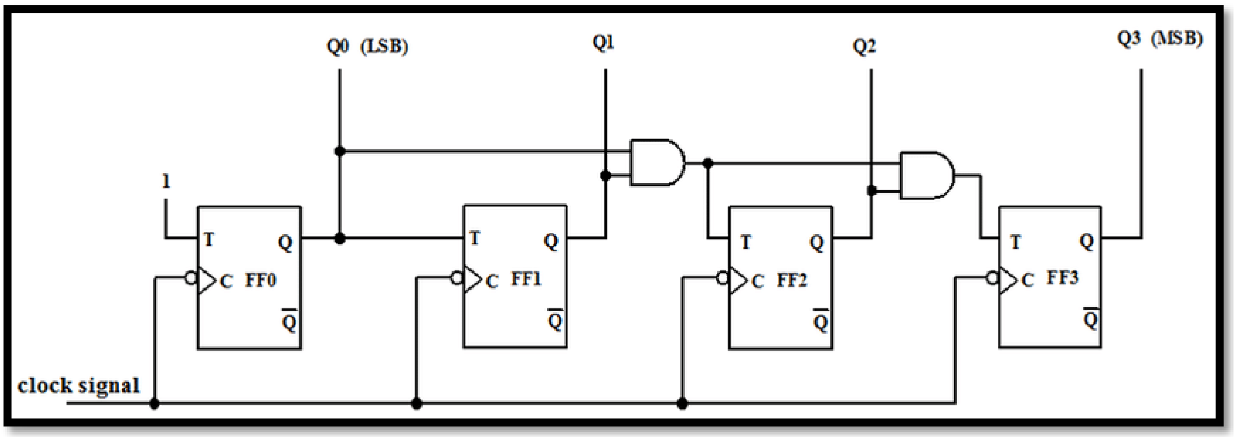


Explanation:-

- To begin with, mode, reset and clock (input by IR sensor)are the inputs given to counter which counts the number of passengers in lift and gives output called counter output. If mode =1 (also called up mode), means that the number of passengers are increasing in the elevator. Similarly if mode =0 (also called down mode), means that the number of passengers are decreasing in the elevator. If reset =1 , the counter will give output as zero
- In addition, multiplier has two inputs , one is the counter output and second is the average weight of single passenger. These two inputs are multiplicands which are multiplied by multiplier producing output in the form of total weight.
- In conclusion, two inputs namely total weight and maximum capacity of elevator are compared using comparator to get final output. If total weight is more than maximum capacity -> final output =1 which shows that elevator is overloaded and led glows . If total weight is less than maximum capacity -> final output =0 which shows that elevator is not overloaded and led does not glow

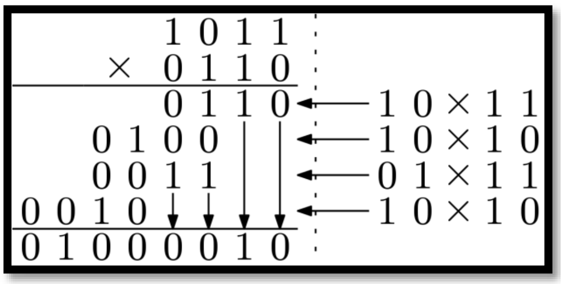
Counters

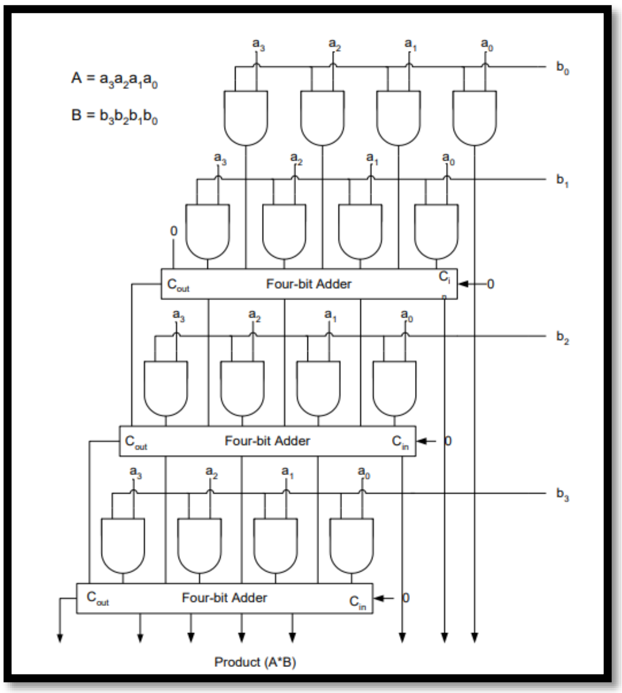
Counters are group of flip-flops. Counters are used to count specific things required as per the need, they are sequential circuits and digital counters counts the pulses, there are many different types of digital counters like Asynchronous Counters, Synchronous Counters, Mod Control Counters, etc. Here we have used 4-bit mod controlled synchronous counter and we also have reset option in it from which we can clear the output of counter or basically the flip-flops used in it. We have made mod controlled counter so the input mode works as when it logic 1 it counts as Up-Counter and for logic 0 it counts as Down-Counter.



Multiplier

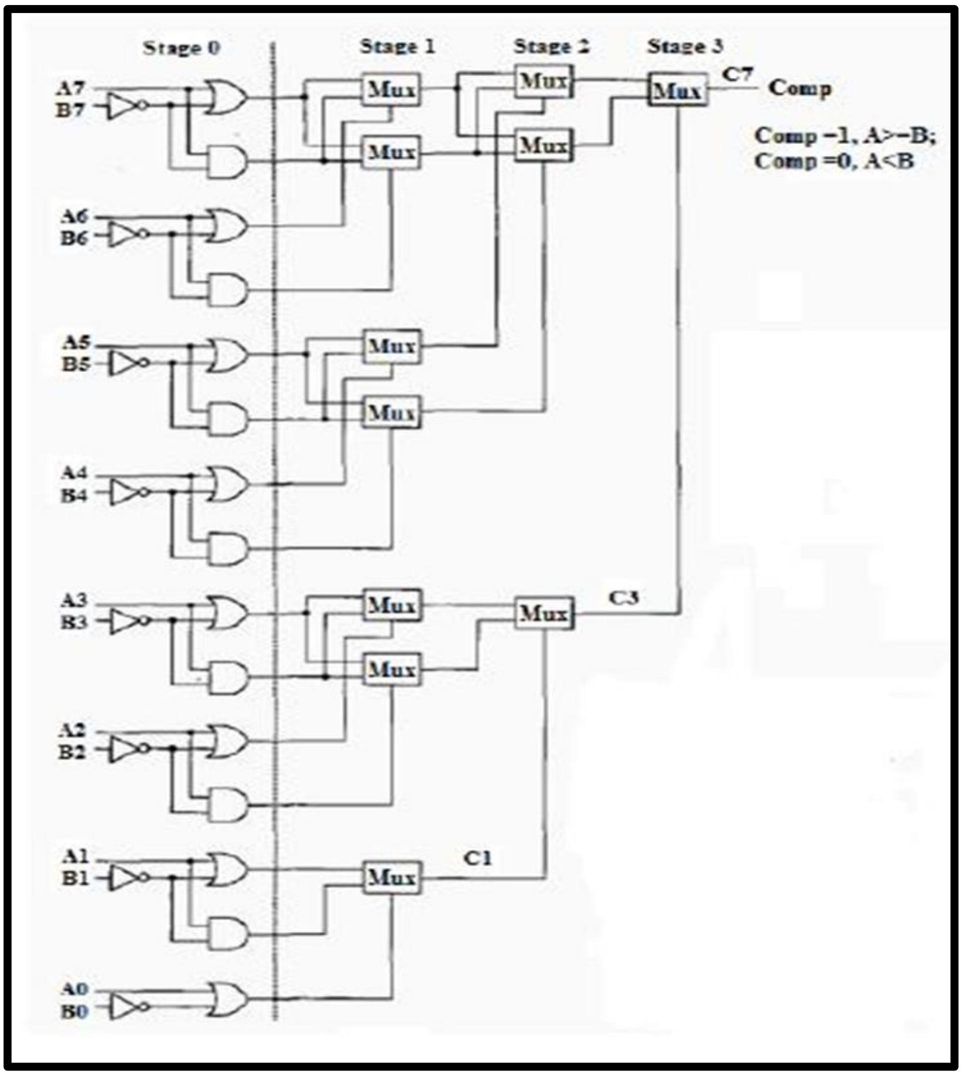
A digital multiplier is a combinational logic circuit used to multiply two numbers. The two numbers which are being multiplied are called multiplicand and the output of the multiplication is called product. They are made with the help of half-adders, full-adders and logic gates. In multipliers first the single bits of multiplier and all the bits of multiplicand are passed by AND gates then same follows for other bits of multiplicand then they are passed to adders and sequential partial products are formed after shifting after each bit of multiplier and then they are passed to adders for addition. The whole process is done as first partial products a0b0, a0b1, a1b0, a1b1 are formed by means of AND gate then adders are used and added by shifting the next bits of each partial products as c0, c1, c2, c3 are generated here with the help of adders .





Comparator

Digital comparators takes two numbers as input and provide the output according to which number is greater or smaller. We manually can check it by calculating the magnitude/decimal equivalent of the numbers, while here we calculate as we start from most significant bit and move towards lowest significant bits of both the numbers while doing this we check if the number are same we move to next low bit or if they are not same then we check which bit is greater and according that number is greater.



Verilog Code

```
// DIGITAL LOGIC DESIGN

// COURSE CODE :- 2EC303

// Title - Elevator Weight Manager

/*

Made by :-

-> 20BEC024 ( Dhruv Dhoalriya )

-> 20BEC026 ( Dhruvi Patel )

*/

// Verilog Code


// Declaration of module

Module
Elevator_Weight_Manager(counter_output,mode,clock,reset,avg_we
ight,total_weight,final_output,max_weight);


//Variable Declaration for Counter

output reg [3:0]counter_output;     //output of counter

input mode,clock,reset;      // If Mode=1 -> up, If Mode = 0 -> down


//Variable Declaration for Multiplier

input [3:0]avg_weight;     //Average weight of person

output [7:0]total_weight;     //total weight of person
```

```
//Variable Declaration for Comparator
output reg final_output;     // Final output of comparator
input [7:0]max_weight;     // Maximum capacity of Elevator

// Code of Counter
Always @(posedge clock)
begin
if(reset)
        counter_output=0;
else
   if(mode)
                         counter_output <= counter_output+1;
  else
                         counter_output <= counter_output-1;
 end

//Total weight Calculation by Multiplier
multiplier M1(counter_output,avg_weight,total_weight);
```

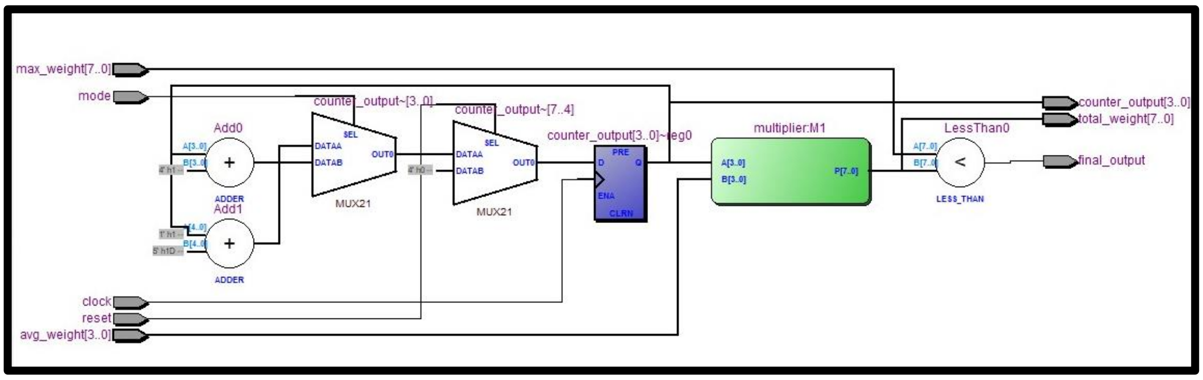
```
// Weigth Comparision
Always @(total_weight or max_weight)
begin
If ( total_weight > max_weight )
      final_output = 1;     // LED will glow because of Overload condition
else
 final_output = 0;     // LED will not glow because of no Overload condition
 end
 Endmodule

//4 bit Multiplier
module multiplier(A, B, P);

//4-bit inputs A and B Multiplicand and Multiplier
input[3:0] A;
input[3:0] B;

//Output P is product of A and B
output[7:0] P;

// Wire for Sum
wire[3:0]  S1;
wire[3:0]  S2;
wire[3:0]  S3;
```

```
// Wire for Carry
wire[3:0]  C1;
wire[3:0]  C2;
wire[3:0]  C3;

// Partial Product Rows
wire[3:0]  AB0;
wire[3:0]  AB1;
wire[3:0]  AB2;
wire[3:0]  AB3;

//  Calculation of Product bits
assign AB0[0] = A[0] & B[0] ;
assign AB1[0] = A[0] & B[1] ;
assign AB0[1] = A[1] & B[0] ;
assign AB1[1] = A[1] & B[1] ;
assign AB0[2] = A[2] & B[0] ;
assign AB1[2] = A[2] & B[1] ;
assign AB0[3] = A[3] & B[0] ;
assign AB1[3] = A[3] & B[1] ;
assign AB2[0] = A[0] & B[2] ;
assign AB3[0] = A[0] & B[3] ;
assign AB2[1] = A[1] & B[2] ;
assign AB3[1] = A[1] & B[3] ;

assign AB2[2] = A[2] & B[2] ;

assign AB3[2] = A[2] & B[3] ;

assign AB2[3] = A[3] & B[2] ;

assign AB3[3] = A[3] & B[3] ;



//Half Adders :-  Used to add the partial products ,when there are
only two inputs A,B


Half_Adder HA1 (AB0[1], AB1[0], C1[0], S1[0]);

Half_Adder HA2 (AB1[3], C1[2], C1[3], S1[3]);

Half_Adder HA3 (S1[1], AB2[0], C2[0], S2[0]);

Half_Adder HA4 (S2[1], AB3[0], C3[0], S3[0]);


//Full Adders :- Used to add the partial products,when there are 3
inputs A,B,Cin

Full_Adder FA1 (AB0[2], AB1[1], C1[0], C1[1], S1[1]);

Full_Adder FA2 (AB0[3], AB1[2], C1[1], C1[2], S1[2]);

Full_Adder FA3 (S1[2], AB2[1], C2[0], C2[1], S2[1]);

Full_Adder FA4 (S1[3], AB2[2], C2[1], C2[2], S2[2]);

Full_Adder FA5 (C1[3], AB2[3], C2[2], C2[3], S2[3]);

Full_Adder FA6 (S2[2], AB3[1], C3[0], C3[1], S3[1]);

Full_Adder FA7 (S2[3], AB3[2], C3[1], C3[2], S3[2]);

Full_Adder FA8 (C2[3], AB3[3], C3[2], C3[3], S3[3]);
```

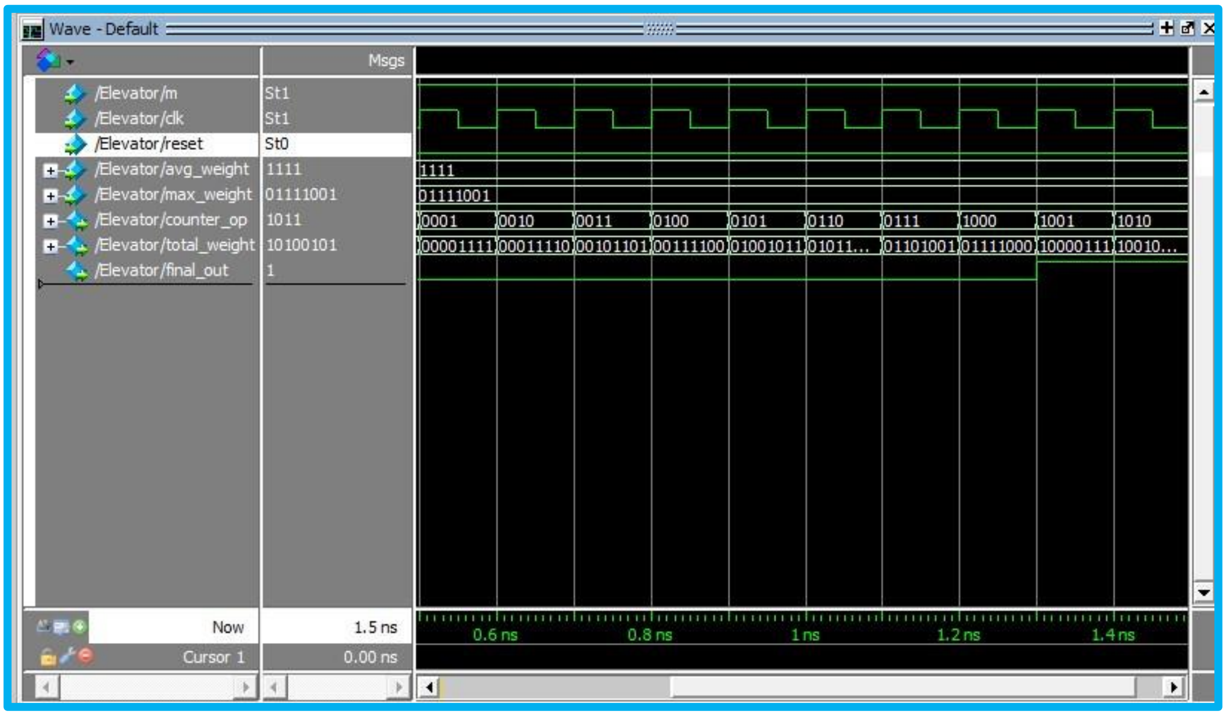
```
//Final output of adders constitute Final product(Multiplication) P
of 8 bits(max)
assign P[0] = AB0[0] ;
assign P[1] = S1[0]  ;
assign P[2] = S2[0]  ;
assign P[3] = S3[0]  ;
assign P[4] = S3[1]  ;
assign P[5] = S3[2]  ;
assign P[6] = S3[3]  ;
assign P[7] = C3[3]  ;
endmodule

//Half Adder Code


module Half_Adder (X, Y, Cout, Sum);
input  X ,Y;
output Cout , Sum;
assign Sum = X ^ Y ;
assign Cout = X & Y ;
endmodule
```

```
//Full Adder Code


module Full_Adder (X, Y, Cin, Cout, Sum);
input  X , Y , Cin;
output Cout , Sum;
assign Sum = X ^ Y ^ Cin ;
assign Cout = (X & Y) | (Y & Cin) | (X & Cin) ;
endmodule
```

RTL Viewer



RTL Stimulation



Conclusion

After successful compliation of code , performing its stimulation we can conclude the we can make a elevator weight manager using a counter , comparator and Multiplier .

References

-> Verilog HDL : A guide to digital design and Synthesis . By Samir Palnitkar .